

CCCCCCCC	SSSSSSSS	PPPPPPPP	WW	WW	AAAAAA	IIIIII	TTTTTT
CCCCCCCC	SS	PP	PP	WW	AA	II	TT
CC	SS	PP	PP	WW	AA	II	TT
CC	SS	PP	PP	WW	AA	II	TT
CC	SSSSSS	PPPPPPPP	WW	WW	AA	II	TT
CC	SSSSSS	PPPPPPPP	WW	WW	AA	II	TT
CC	SS	PP	WW	WW	AAAAAA	II	TT
CC	SS	PP	WW	WW	AAAAAA	II	TT
CC	SS	PP	WWWW	WWWW	AA	II	TT
CC	SS	PP	WWWW	WWWW	AA	II	TT
CCCCCCCC	SSSSSSSS	PP	WW	WW	AA	IIIIII	TT
CCCCCCCC	SSSSSSSS	PP	WW	WW	AA	IIIIII	TT

LL	IIIIII	SSSSSSSS
LL	IIII	SSSSSSSS
LL	II	SS
LL	II	SS
LL	II	SSSSSS
LL	II	SSSSSS
LL	II	SS
LL	II	SS
LLLLLLLL	IIIIII	SSSSSSSS
LLLLLLLL	IIIIII	SSSSSSSS

(3)	76	'CSP\$\$RESUME'
(4)	127	'CSP\$\$WAIT - Asynchronous wait for AST completion.'
(5)	192	'CSP\$\$FORK - create new execution thread'
(6)	262	'CSP\$\$SAVE_STACK - save stack frames prior to suspending thread'
(7)	331	'CSP\$\$CREATE_CTX - allocate and initialize context block'
(8)	363	'CSP\$\$DELETE_CTX - terminate thread'

0000 1
0000 2 .TITLE 'CSPWAIT'
0000 3 .IDENT 'V04-000'
0000 4
0000 5 *****
0000 6 *
0000 7 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 * ALL RIGHTS RESERVED.
0000 10 *
0000 11 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 * TRANSFERRED.
0000 17 *
0000 18 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 * CORPORATION.
0000 21 *
0000 22 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 *
0000 25 *
0000 26 *****
0000 27 *
0000 28 *++
0000 29 *
0000 30 : FACILITY: VMS Cluster Server Process
0000 31 :
0000 32 : ABSTRACT: Subroutines to initiate "wait" states for Cluster Servers
0000 33 : and to create new threads of context.
0000 34 :
0000 35 : AUTHOR: Paul R. Beck
0000 36 :
0000 37 : DATE: 3-MAR-1983 Last Edit: 22-JUN-1983 19:17:35
0000 38 :
0000 39 : REVISION HISTORY:
0000 40 :
0000 41 : V03-005 ADE0004 Alan D. Eldridge 24-Apr-1984
0000 42 : Use CSP\$SCRASH rather than BUG_CHECK.
0000 43 :
0000 44 : V03-004 ADE0003 Alan D. Eldridge 22-Mar-1984
0000 45 : Fix synchronization between CSP\$SWAIT and CSP\$RESUME.
0000 46 :
0000 47 : V03-003 ADE0002 Alan D. Eldridge 28-Feb-1984
0000 48 : Change name of CSP\$SOPCOM to CSP\$TELL_OPCOM
0000 49 :
0000 50 : V03-002 ADE0001 Alan D. Eldridge 3-Dec-1983
0000 51 : Move CSP\$SAVE_STACK to this module from CSP.B32 since BLISS is
0000 52 : not really appropriate for munging the stack. Move CSP\$RESUME,
0000 53 : CSP\$CREATE_CTX, and CSP\$DELETE_CTX as well so that all routines
0000 54 : callable by the client threads are in one module. Changed
0000 55 : synchronization between CSP\$RESUME and CSP\$SWAIT.
0000 56 :
0000 57 :
0000 58 :
0000 59 :
0000 60 :
0000 61 :
0000 62 :
0000 63 :
0000 64 :
0000 65 :
0000 66 :
0000 67 :
0000 68 :
0000 69 :
0000 70 :
0000 71 :
0000 72 :
0000 73 :
0000 74 :
0000 75 :
0000 76 :
0000 77 :
0000 78 :
0000 79 :
0000 80 :
0000 81 :
0000 82 :
0000 83 :
0000 84 :
0000 85 :
0000 86 :
0000 87 :
0000 88 :
0000 89 :
0000 90 :
0000 91 :
0000 92 :
0000 93 :
0000 94 :
0000 95 :
0000 96 :
0000 97 :
0000 98 :
0000 99 :
0000 100 :
0000 101 :
0000 102 :
0000 103 :
0000 104 :
0000 105 :
0000 106 :
0000 107 :
0000 108 :
0000 109 :
0000 110 :
0000 111 :
0000 112 :
0000 113 :
0000 114 :
0000 115 :
0000 116 :
0000 117 :
0000 118 :
0000 119 :
0000 120 :
0000 121 :
0000 122 :
0000 123 :
0000 124 :
0000 125 :
0000 126 :
0000 127 :
0000 128 :
0000 129 :
0000 130 :
0000 131 :
0000 132 :
0000 133 :
0000 134 :
0000 135 :
0000 136 :
0000 137 :
0000 138 :
0000 139 :
0000 140 :
0000 141 :
0000 142 :
0000 143 :
0000 144 :
0000 145 :
0000 146 :
0000 147 :
0000 148 :
0000 149 :
0000 150 :
0000 151 :
0000 152 :
0000 153 :
0000 154 :
0000 155 :
0000 156 :
0000 157 :
0000 158 :
0000 159 :
0000 160 :
0000 161 :
0000 162 :
0000 163 :
0000 164 :
0000 165 :
0000 166 :
0000 167 :
0000 168 :
0000 169 :
0000 170 :
0000 171 :
0000 172 :
0000 173 :
0000 174 :
0000 175 :
0000 176 :
0000 177 :
0000 178 :
0000 179 :
0000 180 :
0000 181 :
0000 182 :
0000 183 :
0000 184 :
0000 185 :
0000 186 :
0000 187 :
0000 188 :
0000 189 :
0000 190 :
0000 191 :
0000 192 :
0000 193 :
0000 194 :
0000 195 :
0000 196 :
0000 197 :
0000 198 :
0000 199 :
0000 200 :
0000 201 :
0000 202 :
0000 203 :
0000 204 :
0000 205 :
0000 206 :
0000 207 :
0000 208 :
0000 209 :
0000 210 :
0000 211 :
0000 212 :
0000 213 :
0000 214 :
0000 215 :
0000 216 :
0000 217 :
0000 218 :
0000 219 :
0000 220 :
0000 221 :
0000 222 :
0000 223 :
0000 224 :
0000 225 :
0000 226 :
0000 227 :
0000 228 :
0000 229 :
0000 230 :
0000 231 :
0000 232 :
0000 233 :
0000 234 :
0000 235 :
0000 236 :
0000 237 :
0000 238 :
0000 239 :
0000 240 :
0000 241 :
0000 242 :
0000 243 :
0000 244 :
0000 245 :
0000 246 :
0000 247 :
0000 248 :
0000 249 :
0000 250 :
0000 251 :
0000 252 :
0000 253 :
0000 254 :
0000 255 :
0000 256 :
0000 257 :
0000 258 :
0000 259 :
0000 260 :
0000 261 :
0000 262 :
0000 263 :
0000 264 :
0000 265 :
0000 266 :
0000 267 :
0000 268 :
0000 269 :
0000 270 :
0000 271 :
0000 272 :
0000 273 :
0000 274 :
0000 275 :
0000 276 :
0000 277 :
0000 278 :
0000 279 :
0000 280 :
0000 281 :
0000 282 :
0000 283 :
0000 284 :
0000 285 :
0000 286 :
0000 287 :
0000 288 :
0000 289 :
0000 290 :
0000 291 :
0000 292 :
0000 293 :
0000 294 :
0000 295 :
0000 296 :
0000 297 :
0000 298 :
0000 299 :
0000 300 :
0000 301 :
0000 302 :
0000 303 :
0000 304 :
0000 305 :
0000 306 :
0000 307 :
0000 308 :
0000 309 :
0000 310 :
0000 311 :
0000 312 :
0000 313 :
0000 314 :
0000 315 :
0000 316 :
0000 317 :
0000 318 :
0000 319 :
0000 320 :
0000 321 :
0000 322 :
0000 323 :
0000 324 :
0000 325 :
0000 326 :
0000 327 :
0000 328 :
0000 329 :
0000 330 :
0000 331 :
0000 332 :
0000 333 :
0000 334 :
0000 335 :
0000 336 :
0000 337 :
0000 338 :
0000 339 :
0000 340 :
0000 341 :
0000 342 :
0000 343 :
0000 344 :
0000 345 :
0000 346 :
0000 347 :
0000 348 :
0000 349 :
0000 350 :
0000 351 :
0000 352 :
0000 353 :
0000 354 :
0000 355 :
0000 356 :
0000 357 :
0000 358 :
0000 359 :
0000 360 :
0000 361 :
0000 362 :
0000 363 :
0000 364 :
0000 365 :
0000 366 :
0000 367 :
0000 368 :
0000 369 :
0000 370 :
0000 371 :
0000 372 :
0000 373 :
0000 374 :
0000 375 :
0000 376 :
0000 377 :
0000 378 :
0000 379 :
0000 380 :
0000 381 :
0000 382 :
0000 383 :
0000 384 :
0000 385 :
0000 386 :
0000 387 :
0000 388 :
0000 389 :
0000 390 :
0000 391 :
0000 392 :
0000 393 :
0000 394 :
0000 395 :
0000 396 :
0000 397 :
0000 398 :
0000 399 :
0000 400 :
0000 401 :
0000 402 :
0000 403 :
0000 404 :
0000 405 :
0000 406 :
0000 407 :
0000 408 :
0000 409 :
0000 410 :
0000 411 :
0000 412 :
0000 413 :
0000 414 :
0000 415 :
0000 416 :
0000 417 :
0000 418 :
0000 419 :
0000 420 :
0000 421 :
0000 422 :
0000 423 :
0000 424 :
0000 425 :
0000 426 :
0000 427 :
0000 428 :
0000 429 :
0000 430 :
0000 431 :
0000 432 :
0000 433 :
0000 434 :
0000 435 :
0000 436 :
0000 437 :
0000 438 :
0000 439 :
0000 440 :
0000 441 :
0000 442 :
0000 443 :
0000 444 :
0000 445 :
0000 446 :
0000 447 :
0000 448 :
0000 449 :
0000 450 :
0000 451 :
0000 452 :
0000 453 :
0000 454 :
0000 455 :
0000 456 :
0000 457 :
0000 458 :
0000 459 :
0000 460 :
0000 461 :
0000 462 :
0000 463 :
0000 464 :
0000 465 :
0000 466 :
0000 467 :
0000 468 :
0000 469 :
0000 470 :
0000 471 :
0000 472 :
0000 473 :
0000 474 :
0000 475 :
0000 476 :
0000 477 :
0000 478 :
0000 479 :
0000 480 :
0000 481 :
0000 482 :
0000 483 :
0000 484 :
0000 485 :
0000 486 :
0000 487 :
0000 488 :
0000 489 :
0000 490 :
0000 491 :
0000 492 :
0000 493 :
0000 494 :
0000 495 :
0000 496 :
0000 497 :
0000 498 :
0000 499 :
0000 500 :
0000 501 :
0000 502 :
0000 503 :
0000 504 :
0000 505 :
0000 506 :
0000 507 :
0000 508 :
0000 509 :
0000 510 :
0000 511 :
0000 512 :
0000 513 :
0000 514 :
0000 515 :
0000 516 :
0000 517 :
0000 518 :
0000 519 :
0000 520 :
0000 521 :
0000 522 :
0000 523 :
0000 524 :
0000 525 :
0000 526 :
0000 527 :
0000 528 :
0000 529 :
0000 530 :
0000 531 :
0000 532 :
0000 533 :
0000 534 :
0000 535 :
0000 536 :
0000 537 :
0000 538 :
0000 539 :
0000 540 :
0000 541 :
0000 542 :
0000 543 :
0000 544 :
0000 545 :
0000 546 :
0000 547 :
0000 548 :
0000 549 :
0000 550 :
0000 551 :
0000 552 :
0000 553 :
0000 554 :
0000 555 :
0000 556 :
0000 557 :
0000 558 :
0000 559 :
0000 560 :
0000 561 :
0000 562 :
0000 563 :
0000 564 :
0000 565 :
0000 566 :
0000 567 :
0000 568 :
0000 569 :
0000 570 :
0000 571 :
0000 572 :
0000 573 :
0000 574 :
0000 575 :
0000 576 :
0000 577 :
0000 578 :
0000 579 :
0000 580 :
0000 581 :
0000 582 :
0000 583 :
0000 584 :
0000 585 :
0000 586 :
0000 587 :
0000 588 :
0000 589 :
0000 590 :
0000 591 :
0000 592 :
0000 593 :
0000 594 :
0000 595 :
0000 596 :
0000 597 :
000

CSPWAIT
V04-000

G 9

16-SEP-1984 01:10:46 VAX/VMS Macro V04-00
5-SEP-1984 04:09:09 [SYSLOA.SRC]CSPWAIT.MAR;1

Page 2
(1)

0000 58 :
0000 59 :
0000 60 :--

V03-001 PRB0205 Paul Beck
Change CTX\$ symbols to CLX\$ to prevent conflict with RCP.
6-JUN-1983

CSP
VAX

Sym
Pas
Sym
Pse
Cro
Ass

The
100
The
393
12

Mac

-\$2
-\$2
-\$2
TOT
155
The
MAC

```
0000 62 : Include files
0000 63 : Include files
0000 64 :
0000 65     $SFDEF
0000 66     $CLXDEF
0000 67
0000 68 :
0000 69 : Own storage
0000 70 :
00000001 0000 71 CONTEXT_ID:    :LONG 1
00000140 0004 72 CLX_SIZE:    :LONG CLX$K_LENGTH
0008 73
0008 74 : Storage for next CLX index
          : CLX length used as input an parameter
          : when 'call by reference' is needed
```

'CSP\$\$RESUME'

0008	76	SBTTL 'CSP\$\$RESUME'	
0008	77	++	
0008	78		
0008	79	Completion of AST for asynchronous calls. Reschedule the thread. This	
0008	80	routine can be specified directly as an AST, or it may be called from	
0008	81	another AST.	
0008	82		
0008	83	CALLING SEQUENCE: Standard AST (or called from and AST jacket routine).	
0008	84	May also be called from "normal" level	
0008	85		
0008	86	FORMAL PARAMETERS: P1 = address of thread's context block	
0008	87		
0008	88	COMPLETION CODES: N/A	
0008	89		
0000	90	--	
0008	91	ENTRY CSP\$\$RESUME, 0 ; Save nothing	
000A	92		
000A	93		
000A	94	This routine, since it is most often called via an AST, can	
000A	95	come before or may interrupt the execution of CSPSSWAIT.	
000A	96		
000A	97		
07 0B A0 01	50 04 AC	D0 E3 000A 98 MOVL 4(AP),R0	
		000E 99 BBCS #CLX\$V_MUTEX,CLX\$B_FLAGS(R0),10\$; If BS, blocked by CSPSSWAIT	
		0013 100	
		0013 101	
		0013 102	
		0013 103	
		0013 104	
22 0B A0 1F	02 00	E2 0013 105 BBSS #CLX\$V_RESUME_REQ -	
		0015 106 CLX\$B_FLAGS(R0),70\$; Tell CSPSSWAIT we interrupted	
		11 0018 107 BRB 50\$; its execution	
		001A 108 10\$: ; Done	
		001A 109	
		001A 110	
		001A 111	
		001A 112	
03 0B A0 14	00 50	E3 001A 113 BBCS #CLX\$V_QUEUED,CLX\$B_FLAGS(R0),30\$; If BC, not yet queued	
00000004'FF	60 60	001F 114 REMQUE (R0),R0 ; Remove block from old queue	
14 0B A0 01	01 0E	0022 115 30\$: INSQUE (R0),@CSP\$GQ_RESUME+4 ; ...and reschedule the thread	
		0029 116 BBCC #CLX\$V_MUTEX,CLX\$B_FLAGS(R0),90\$; Release interlock	
		002E 117 SWAKE_S ; Wake the CSP for processing	
		04 0039 118 50\$: RET ; Done	
		003A 119	
00000004'8F	06 06	DD 003A 120 70\$: PUSHL #SS\$-NOPRIVSTRT+4 ; RESUME_REQ should have been 0	
00000008'8F	01 00	11 0040 121 BRB 100\$; MUX should have been set	
00000000'EF	01 FB	DD 0042 122 90\$: PUSHL #SS\$ NOPRIVSTRT+8 ; Report bug	
		0048 123 100\$: CALLS #1,CSP\$\$CRASH ; Should never get here	
		004F 124 HALT	
		0050 125	

CSPSSWAIT - Asynchronous wait for AST co 5-SEP-1984 04:09:09 [SYSLOA.SRC]CSPWAIT.M

				0050	127	SBTTL	CSP\$SWAIT - Asynchronous wait for AST completion.	
				0050	128	++		
				0050	129			
				0050	130	The current stack is saved in an allocated block, which is saved in the		
				0050	131	current thread's context block. A test is then done to see if the completion		
				0050	132	AST completed prior to this routine; if so, the context is rescheduled. The		
				0050	133	routine then forces a scheduler run by collapsing the stack and returning.		
				0050	134	CALLING SEQUENCE: CALL - never called from an AST routine.		
				0050	135	INPUT PARAMETERS: none		
				0050	136	OUTPUT PARAMETERS: N/A		
				0050	137	COMPLETION CODES: N/A		
				0050	138	--		
			OFFC	0050	144	ENTRY CSP\$SWAIT,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; save all registers		
				0052	145			
				0052	146			
				0052	147	Save the current thread context. The context is now on the stack		
				0052	148	including all registers except R0 and R1. R0 and R1 are saved		
				0052	149	separately since the calling standard enforced by the VAX		
				0052	150	architecture does not allow saving them in the entry mask.		
				0052	151			
				0052	152			
54	00000000'GF	16	0052	153	JSB	G^CSP\$SAVE_STACK	; Save stack in current CLX	
	00000000'GF	D0	0058	154	MOVL	G^CSP\$GL_CURCTX,R4	; Get current CLX	
	00000000'GF	D4	005F	155	CLRL	G^CSP\$GL_CURCTX	; This thread no longer active	
				0065	156			
				0065	157			
				0065	158	We must test for a race condition with the completion AST.		
				0065	159			
				0065	160	Since this routine is never called from AST level, it may be		
				0065	161	interrupted by CSP\$RESUME -- but never vice versa.		
				0065	162			
				0065	163			
	28 0B A4 01	E2	0065	164	BBSS	#CLX\$V_MUTEX, CLX\$B_FLAGS(R4),90\$; If BS, interlocked	
	07 0B A4 00	E2	C06A	165	BBSS	#CLX\$V_QUEUED,CLX\$B_FLAGS(R4),50\$; If BS, RESUME occurred	
	00000004'FF	64	UE	006F	166			
				006F	167	INSQUE (R4),ACSP\$GQ_WAIT+4	; before we interlocked	
				0076	168	50\$:	; Queue it to wait list	
				0076	169			
				0076	170			
				0076	171	Return to the scheduler. This is done by collapsing the stack to a		
				0076	172	known point, where there is a call frame used to enter the		
				0076	173	scheduler. Then return with a success code. This will cause the		
				0076	174	scheduler to be reentered.		
				0076	175			
	1A 0B A4 01	E5	0076	176	BBCC	#CLX\$V_MUTEX,CLX\$B_FLAGS(R4),90\$; Release interlock	
	0A 0B A4 02	E5	007B	177	BBCC	#CLX\$V_RESUME_REQ,CLX\$B_FLAGS(R4),70\$; If BS, RESUME occurred	
				0080	178			
	54 64	OF	0080	179	REMQUE (R4),R4		; since we interlocked	
	00000004'FF	64	OE	0083	180	INSQUE (R4),ACSP\$GQ_RESUME+4	; Remove CLX from WAIT	
	SD 00000000'GF	DD	008A	181	70\$:	MOVL G^CSP\$GL_BASE_FP,FP	; Que it to RESUME list	
				0091	182		; Point to scheduler	
	50 00'	DD	0091	183	MOVL S#SS\$NORMAL,RO		; stack frame	
							; Declare success	

CSPSSWAIT - Asynchronous wait for AST co 16-SEP-1984 01:10:46 VAX/VMS Macro V04-00
5-SEP-1984 04:09:09 [SYSLOA.SRC]CSPWAIT.MAR;1 Page 6 (4)

04 0094 184 RET : Go reschedule.
00000000C'8F 0095 185
00000000'EF 01 DD 0095 186 90\$: PUSHL #SS\$ NOPRIVSTRT+12
FB 0098 187 CALLS #1,C5PSS\$CRASH : Use phoney status
00 00A2 188 HALT : report MUTEX conflict
00A3 189
00A3 190 .dsabl lsb : Should never get here

00A3 192 .SBTTL CSP\$SFORK - create new execution thread
 00A3 193 ++
 00A3 194
 00A3 195 This is a fork routine. A new context block is allocated and initialized,
 00A3 196 and the current context is saved and queued to the thread resume (grant)
 00A3 197 queue. The stack is NOT reclaimed, and the scheduler is NOT called. When the
 00A3 198 scheduler is eventually entered, each thread thus queued is resumed at the
 00A3 199 return from this routine. The completion code is used to determine whether
 00A3 200 the execution context is the new thread (SS\$NORMAL) or simply the creator
 00A3 201 of the thread (0). For example:
 00A3 202
 00A3 203 CALLS #0,CSP\$SFORK
 00A3 204 BLBC R0, 10\$; continue executing old thread
 00A3 205 BRW NEW_THREAD ; start executing new thread
 00A3 206
 00A3 207
 00A3 208
 00A3 209
 00A3 210
 00A3 211 When creating a thread this way, be aware that the context saved
 00A3 212 is in the registers and stack. Local variables should be so defined.
 00A3 213
 00A3 214 CALLING SEQUENCE: CALL
 00A3 215
 00A3 216 FORMAL PARAMETERS: none
 00A3 217
 00A3 218 COMPLETION CODES:
 00A3 219
 00A3 220 SS\$NORMAL = The new thread has been resumed by the scheduler
 00A3 221 0 = The thread has been queued, context is intact
 00A3 222 --
 OFFC 00A3 223 .entry CSP\$SFORK,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; save all registers
 00000000'GF DD 00A5 224
 00000000'GF DD 00A5 225 PUSHL G^CSP\$GL_CURCTX ; Temp save current context
 00000000'GF DD 00AB 226
 00000000'GF DD 00AB 227
 00000000'GF DD 00AB 228 First, create the context block.
 00000000'GF DD 00AB 229
 00000000'GF DD 00AB 230
 52 50 00 00 00B1 231 JSB G^CSP\$CREATE_CTX ; Allocate new CLX block
 1A 13 00 00 00B4 232 MOVL R0,R2 ; Copy of new CLX pointer
 00B6 233 BEQL 10\$; If EQL, allocation failure
 00B6 234
 00B6 235
 00B6 236 Now, save the current stack in the context block. This requires
 00B6 237 faking out CSP\$SAVE_STACK as to which is the current context.
 00B6 238
 00B6 239
 00000000'GF 52 DD 00B6 240 MOVL R2,G^CSP\$GL_CURCTX ; Store new context pointer
 00000000'GF 16 DD 00B0 241 JSB G^CSP\$SAVE_STACK ; Save stack in new CLX
 00C3 242
 00C3 243
 00C3 244
 00C3 245
 00C3 246
 00C3 247
 00C3 248 Next, schedule it. Force the saved R0 to SS\$NORMAL so that
 upon rescheduling the caller will sense that this is the fork
 thread executing.

CSP\$\$FORK - create new execution thread

28 A2 00'	DD 00C3 249	MOVL S^#SS\$NORMAL,CLXSL_R0(R2)	; "jam" success code
52 FF3A CF 01	DD 00C7 250	PUSHL R2	; Setup parameter
	FB 00C9 251	CALLS #1,CSP\$\$RESUME	; ...and reschedule the thread
	00CE 252	:	
	00CE 253	:	
	00CE 254	Return with status=0 indicating that this isn't the fork thread yet.	
	00CE 255	:	
	00CE 256	:	
00000000'GF 8ED0 04	D4 00CE 257	CLRL R0	; "SS\$NOT_FORK"
	00D0 258 10\$:	POPL G^CSP\$GL_CURCTX	; Restore current CLX pointer
	00D7 259	RET	
	00D8 260	:	

'CSPSSSAVE_STACK - save stack frames pri 16-SEP-1984 01:10:46 VAX/VMS Macro V04-00
5-SEP-1984 04:09:09 [SYSLOA.SRC]CSPWAIT.MAR;1 Page 9 (6)

00D8 262 SBTTL 'CSPSSSAVE_STACK - save stack frames prior to suspending thread'
 00D8 263 ++
 00D8 264
 00D8 265 Allocate memory to save the current stack (from top to the scheduler call
 00D8 266 frame), and store it in the current thread's context block.
 00D8 267
 00D8 268 CALLING SEQUENCE: JSB
 00D8 269
 00D8 270 FORMAL PARAMETERS: none
 00D8 271
 00D8 272 IMPLIED INPUTS: CSP\$GL_CURCTX = address of context block in which to store
 00D8 273 the saved stack.
 00D8 274
 00D8 275 COMPLETION CODES: SSS_NORMAL, or failure code from LIBSS\$GET_VM
 00D8 276
 00D8 277 --
 00D8 278
 00000000 279 .PSECT SPLIT\$, NOWRT, NOEXE, 2
 00000000 280
 281 P.AAK: .ASCII 'CSPSSSAVE_STACK: CURCTX=0'<0><0><0> ;
 001C 282 P.AAJ: .LONG 17694745 ;
 00000000 283 .ADDRESS P.AAK ;
 0024 284
 00000000 285 .PSECT SCODE\$, NOWRT, 2
 0000 286
 007C 8F 287 CSPSSSAVE_STACK::
 BB 288 PUSHR #^M<R2,R3,R4,R5,R6> ; Save regs
 0000 289
 52 00000000'EF 290 MOVL CSP\$GL_CURCTX, R2 ; Get current CLX block
 11 12 000B 291 BNEQ 10\$; If NEQ, it's there
 0000001C'EF 9F 000D 292 PUSHAB P.AAJ ; Setup message desc.
 00000000'EF 01 FB 0013 293 CALLS #1, CSP\$TELL_OPCOM ; Display message
 50 D4 001A 294 CLRL R0 ; Indicate error
 3F 11 001C 295 BRB 40\$; Take common exit
 001E 296
 10\$: 001E 297 :
 001E 298 :
 001E 299 : We save the stack from CSP\$GL_BASE_FP up to and including the
 001E 300 : current stack frame (note JSB interface). This assumes that this
 001E 301 : routine is always called from a WAIT or a FORK routine which has
 001E 302 : been CALL'ed by the thread which needs the context block saved.
 001E 303
 001E 304
 001E 305 ASSUME CLXSL R1 EQ 4+CLXSL_R0
 28 A2 50 7D 001E 306 MOVQ R0, CLXSL_R0(R2) ; Save R0,R1
 08 A2 08 8A 0022 307 BICB #CLX\$M_LOCAL_STACK, CLX\$B_FLAGS(R2) ; Init flag
 00000100 8F 38 A2 5D C3 0026 308 SUBL3 FP, G^CSP\$GL_BASE_FP, CLX\$C_STACKSIZE(R2) ; Determine stack size
 38 A2 D1 002F 309 CMPL CLXSL_STACKSIZE(R2), #CLX\$R_LOCAL_STACK ; Overflow CLX?
 08 1A 0037 310 BGTRU 20\$; If GTRU, yes
 3C A2 40 A2 9E 0039 311 MOVAB CLX\$B_LOCAL_STACK(R2), CLX\$A_STACK(R2) ; Setup stack ptr
 08 A2 08 88 003E 312 BISB #CLX\$A_LOCAL_STACK, CLX\$B_FLAGS(R2) ; Indicate CLX stack
 10 11 0042 313 BRB 30\$; Continue
 0044 314 20\$: :
 0044 315 :
 0044 316 : Must allocate a block to hold the stack

3C A2 9F	0044	317	:			
38 A2 9F	0044	318		PUSHAB	CLXSA_STACK(R2)	: Point to block ptr
02 FB	0047	319		PUSHAB	CLXSL_STACKSIZE(R2)	: Point to block size
09 50 E9	004A	320		CALLS	#2,G^IBSGET_VM	: Allocate the block
3C B2 6D	0051	321		BLBC	R0,40\$: If LBC, failed
38 A2 28	0054	322		MOVC3	CLXSL_STACKSIZE(R2),(FP), -	: Copy stack
	005A	323	30\$:		@CLXSA_STACK(R2)	
	005A	324				
	005A	325				
50 01 D0	005A	326		MOVL	#1, R0	: Setup success status
007C 8F	005D	327	40\$:	POPR	#^M<R2,R3,R4,R5,R6>	: Restore regs
05 BA	0061	328		RSB		: Done
	0062	329				

'CSP\$CREATE_CTX - allocate and initialize context block'

0062 331 .SBTTL 'CSP\$CREATE_CTX - allocate and initialize context block'
 0062 332 :++
 0062 333 :
 0062 334 : Allocate and initialize a context block.
 0062 335 :
 0062 336 : CALLING SEQUENCE: JSB
 0062 337 :
 0062 338 : INPUT PARAMETERS: R0 Scratch
 0062 339 :
 0062 340 : OUTPUT PARAMETERS: R0 Address of context block is returned
 0062 341 : (0 if error).
 0062 342 :
 0062 343 :--
 0062 344 :CSP\$CREATE_CTX::
 3F BB 0062 345 PUSHR #^M<R0,R1,R2,R3,R4,R5> ; Save regs
 0064 346 :
 00000004'EF 6E 9F 0064 347 PUSHAB (SP) ; Address of block pointer
 00000000'GF 02 FB 0066 348 PUSHAB CLX_SIZE ; Address of block length
 00 BE 0140 8F 00 6E 00 2C 0076 349 CALLS #2,G^LIB\$GET_VM ; Allocate the block
 24 A0 00000000'EF 50 6E 00 2C 0076 350 :
 00000000'EF 02 D0 007F 351 BLBC R0,10\$; If LBC, then failed
 00000000'EF 02 D0 0082 352 MOVC5 #0,(SP),#0,#CLX\$K_LENGTH,@(SP) ; Zero it
 00000000'EF 02 D6 008A 353 MOVL (SP),R0 ; Pickup the block
 00000000'EF 02 11 0090 354 MOVL CONTEXT_ID,CLX\$L_INDEX(R0) ; Enter the i.d.
 00000000'EF 02 D4 0092 355 INCL CONTEXT_ID ; Bump the i.d. for next time
 00000000'EF 02 0094 356 BRB 20\$; Take common exit
 00000000'EF 02 0094 357 10\$: CLRL (SP) ; Say "no block allocated"
 3F BA 0094 358 RSB :
 05 0096 359 20\$: POPR #^M<R0,R1,R2,R3,R4,R5> ; Restore regs
 0097 360 :
 0097 361 RSB ; Done

0097	363	.SBTTL 'CSP\$DELETE_CTX - terminate thread'			
0097	364	:++			
0097	365	:			
0097	366	Terminate an execution thread by deleting the context block and			
0097	367	clearing the pointer.			
0097	368	:			
0097	369	CALLING SEQUENCE: JSB			
0097	370	:			
0097	371	FORMAL PARAMETERS: None			
0097	372	:			
0097	373	COMPLETION CODES: N/A			
0097	374	:			
0097	375	--			
00000000'GF	9F	0097	376	CSP\$DELETE_CTX::	
		009D	377	PUSRAB G^CSP\$GL_CURCTX	
		009D	378	;	
	00	BE	009D	379	TSTL a(SP)
	09	12	00A0	380	BNEQ 10\$
00000000'EF	00	FB	00A2	381	CALLS #0 MUMBLE
	0F	11	00A9	382	BRB 20\$
			00AB	383	;
			00AB	384	10\$: PUSHL (SP)
00000004'EF	6E	DD	00AB	385	PUSHAB CLX_SIZE
00000000'GF	02	FB	00AD	386	CALLS #2,G^LIB\$FREE_VM
			00B3	387	;
			00BA	388	20\$: CLRL a(SP)+
	9E	D4	00BA	389	RSB
	05	00BC	00BD	390	;

CSPWAIT
V04-000

E 10

'CSP\$DELETE_CTX - terminate thread'

00BD 392
00BD 393 .end

16-SEP-1984 01:10:46 VAX/VMS Macro V04-00
5-SEP-1984 04:09:09 [SYSLOA.SRC]CSPWAIT.MAR;1

Page 13
(10)

DIS
V04

CSPWAIT
Symbol table

CLX\$A_STACK	= 0000003C
CLX\$B_FLAGS	= 00000008
CLX\$B_LOCAL_STACK	= 00000040
CLX\$K_LENGTH	= 00000140
CLX\$K_LOCAL_STACK	= 00000100
CLX\$L_INDEX	= 00000024
CLX\$L_R0	= 00000028
CLX\$L_R1	= 0000002C
CLX\$L_STACKSIZE	= 00000038
CLX\$M_LOCAL_STACK	= 00000008
CLX\$V_MUTEX	= 00000001
CLX\$V_QUEUED	= 00000000
CLX\$V_RESUME_REQ	= 00000002
CLX_SIZE	00000004 R 01
CONTEXT_ID	00000000 R 01
CSP\$SCRASH	***** X 01
CSP\$CREATE_CTX	00000062 RG 04
CSP\$DELETE_CTX	00000097 RG 04
CSP\$FORK	000000A3 RG 01
CSP\$RESUME	00300008 RG 01
CSP\$SAVE_STACK	00000000 RG 04
CSP\$WAIT	00000050 RG 01
CSP\$GL_BASE_FP	***** X 01
CSP\$GL_CURCTX	***** X 01
CSP\$GQ_RESUME	***** X 01
CSP\$GQ_WAIT	***** X 01
CSP\$TELL_OPCOM	***** X 04
LIB\$FREE_VM	***** X 04
LIB\$GET_VM	***** X 04
MUMBLE	***** X 04
P.AAJ	0000001C R 03
P.AAK	00000000 R 03
SS\$NOPRIVSTRT	***** X 01
SS\$NORMAL	***** X 01
SYSSWAKE	***** GX 01

-----+
! Psect synopsis !
-----+

PSECT name

	Allocation	PSECT No.	Attributes
ABS	00000000 (0.) 00 (0.) NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE		
BLANK	000000D8 (216.) 01 (1.) NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE		
\$ABSS	00000000 (0.) 02 (2.) NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE		
SPLITS	00000024 (36.) 03 (3.) NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG		
SCODES	000000BD (189.) 04 (4.) NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC LONG		

-----+
! Performance indicators !
-----+

Phase

	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.03	00:00:01.28
Command processing	107	00:00:00.45	00:00:03.74
Pass 1	164	00:00:01.40	00:00:06.52

Symbol table sort	0	00:00:00.07	00:00:00.33
Pass 2	85	00:00:00.60	00:00:03.10
Symbol table output	5	00:00:00.03	00:00:00.03
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	394	00:00:02.60	00:00:15.02

The working set limit was 1350 pages.

10006 bytes (20 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 95 non-local and 18 local symbols.
393 source lines were read in Pass 1, producing 25 object records in Pass 2.
12 pages of virtual memory were used to define 11 macros.

```
-----+  
! Macro library statistics !  
-----+
```

Macro library name	Macros defined
\$255\$DUA28:[SYSLOA.OBJ]CLUSTER.MLB;1	1
\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	0
\$255\$DUA28:[SYSLIB]STARLET.MLB;2	7
TOTALS (all libraries)	8

MACRO/LIS=LISS:CSPWAIT/OBJ=OBJ\$CSPWAIT MSRC\$CSPWAIT/UPDATE=(ENH\$CSPWAIT)+EXECMLS/LIB+LIB\$CLUSTER/LIB

0394 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

CSPOPCOM
LIS

CSPWAIT
LIS

CSPRPCAC
LIS

DISTRKI
LIS

CSPCURES
LIS

CSPQUORUM
LIS

CSPMOUNT
LIS

CSPVECTOR
LIS

DSTRLOCK
LIS

CSPCLIENT
LIS

DSTRLOCK
LIS